

[INICIAR SESIÓN](#)[NUESTROS PLANES](#)[TODOS LOS CURSOS](#)[FORMACIONES](#)[CURSOS](#)[PARA EMPRESAS](#)[ARTÍCULOS DE TECNOLOGÍA > PROGRAMACIÓN](#)

Intercambiando caracteres de una String en Java



mario.alvial

26/10/2020

Tenemos un registro que recibe datos como número de NIT y el código postal del usuario, pero estamos teniendo ciertos problemas en nuestro sistema, ya que todos los usuarios registrados están ingresando sus datos de manera diferente, como se muestra en la siguiente tabla:

Nombre: Yuri

NIT: 087824030-66

CÓDIGO POSTAL: 689.016-60

Nombre: Mario

NIT: 959,562.590-60

CÓDIGO POSTAL: 58030-280

El NIT de cada usuario se envía en varios formatos y nuestro sistema está programado para recibir solo los dígitos del documento, sin ningún "." o "-".

Algunas direcciones no se encuentran, porque para buscar la dirección del usuario es necesario que el código postal no tenga ningún formato, es decir, solo tenga los dígitos.

Necesitamos estandarizar esto para que todos se salven de la misma manera.

Para ello crearemos un método que toma los datos del usuario y los formatea, eliminando todo lo innecesario, antes de que sean enviados a alguna lógica de negocio.

Todos los datos que recibiremos serán del tipo `String`, así que creemos un método que reciba una `String`, lo formatea y lo devuelve:

```
public static String formateaDatos(String dato){  
    //lógica de formato aquí  
    return dato;  
}
```

Comencemos con el NIT. Un NIT se compone de 11 dígitos, que pueden estar separados por "." y "-". Entonces, ¿cómo eliminamos estos caracteres no deseados?

Método `replaceAll()`

Necesitamos deshacernos de los caracteres no deseados sin quitar ningún dígito o cambiar el orden de los caracteres en nuestro `String`. Para eso podemos usar el método [replaceAll\(\)](#) de clase `String`.

Este método nos permite reemplazar todas las ocurrencias del carácter que pasamos por otro carácter.

Así que primero vamos a eliminar todos los ".". Así tenemos:

```
public static String formateaDatos(String dato){  
    dato = dato.replaceAll(".", "");  
    return dato;  
}
```

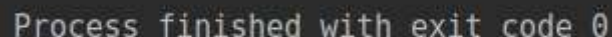
Tenga en cuenta que el primer argumento que pasamos fue ".", porque es el carácter que queremos reemplazar. Después, pasamos una `String` vacía representada por las comillas dobles juntas.

Con eso estamos diciendo que cualquier punto que encuentre en nuestra `String` será reemplazado por nada, por lo que será eliminado.

Para probar, llamemos a este método insertando cualquier NIT:

```
public static void main(String[] args) {  
    System.out.println(formateaDatos("157.108.950-08"));  
}
```

Como resultado tenemos:



```
Process finished with exit code 0
```

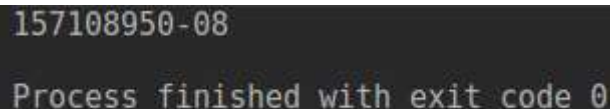
Nada... ¿Dónde está nuestra String formateada? Esto ocurrió porque el `replaceAll()` también acepta expresiones regulares (regex).

En este caso, consideró el punto como una expresión regular. Este regex representa cualquier carácter excepto `\n`, que representa una nueva línea. Con eso todos los caracteres de nuestra String se han eliminado.

Necesitamos hacer con que Java interprete el punto como solo un punto y no como un regex, para eso agregaremos dos barras invertidas antes del punto, esto indica al compilador que el punto tiene un significado diferente al estándar, que en este caso, es un regex. Pronto considerará el punto como un carácter normal, así:

```
public static String formateaDatos(String dato){  
    dato = dato.replaceAll("\\\\.", "");  
    return dato;  
}
```

Probemos de nuevo y veamos qué pasa:



```
157108950-08  
Process finished with exit code 0
```

Lo hicimos. Ahora hagamos lo mismo con `-`. Esto es más simple porque el guión no coincide con ningún regex. Entonces Java no se confundirá. Solo tenemos que poner un `replaceAll()` más:

```
public static String formateaDatos(String dato){  
    dato = dato.replaceAll("\\\\.", "");  
    dato = dato.replaceAll("-", "");  
}
```

```
    return dato;  
}
```

Probemos con el mismo NIT:



```
15710895008  
Process finished with exit code 0
```

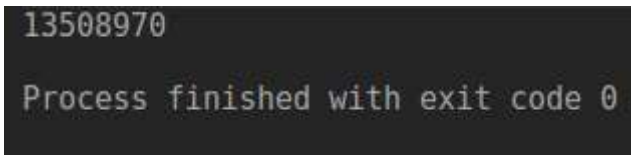
Listo, logramos formatear nuestro NIT y también formateamos el CP.

Si pensamos en un CP tiene esta estructura: "13508-970". Entonces cuando entra en el segundo `replaceAll()` se eliminará el guión.

Probemos esta teoría insertando un CP verdadero:

```
public static void main(String[] args) { System.out.println(formateaDatos("13508-970")); }
```

Y en nuestra consola:



```
13508970  
Process finished with exit code 0
```

Problemas con nuestra resolución

Parece estar bien, ¿verdad? Pero un usuario es imprevisible. Por ejemplo, ¿qué pasaría si envío un código postal con "/" o alguna letra? Veamos:

```
public static void main(String[] args) {  
    System.out.println(formateaDatos("13/508-970"));  
}
```

Resultado:



```
13/508970  
Process finished with exit code 0
```

No manejamos este caso, ¿qué pasa si el dato contiene letras? ¿Tiene otro símbolo extraño? Tendremos que hacer un `replaceAll()` para cada carácter que pueda burlar

nuestro formato?

Date cuenta de que nuestro plan se está volviendo inviable. Es muy difícil poder cubrir todos los casos que pueden romper nuestro método de formateo. Quizás estemos pensando mal.

Lidiando con datos imprevisibles

¿Qué tal si en lugar de decir los caracteres que no permitimos, decimos los caracteres que permitimos para nuestro método `replaceAll()`? Esto eliminaría todo lo que fuera innecesario o inválido.

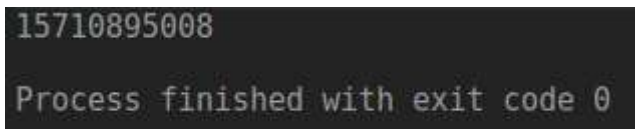
Pensando en un NIT o un CP, ¿qué permiten tener? Solo dígitos, ¿correcto? Entonces, hablemos con el `replaceAll()` que queremos reemplazar todo lo que no sea un dígito por nada, es decir, queremos quitar todo lo que no sea un dígito:

```
''' public static String formateaDatos(String dato){ return dato.replaceAll("[^0-9]+", ""); }  
'''
```

Este regex representa cualquier cosa que no sea un número del 0 al 9. Para probar, insertaremos un NIT lleno de errores, con ".", "-", "/" y letras:

```
público estático vacío principal(String [] args) {  
    System.out.println (formateaDatos ("15 / * 7.1DAS08.950-08 / A?% D"));  
}
```

Nuestro método nos devuelve:



```
15710895008  
Process finished with exit code 0
```

¡Funcionó! Pudimos hacer nuestro formato de método correctamente.

Conclusión

El formateo de datos es muy común. Se debe tener cuidado al tratar con datos que provienen del usuario, ya que nunca se sabe lo que envía el usuario. Además, al involucrarse en un problema como este, donde cada vez es más difícil crear o hacer el

mantenimiento en una funcionalidad, es muy válido detenerse y ver el problema desde otro ángulo, como fue nuestro caso.

Es muy importante saber construir código limpio y poder percibir un problema en nuestro propio código. En [Alura](#) tenemos toda una formación con varios cursos de **Java**, desde lo más básico con el lenguaje hasta conceptos más avanzados como clases, poliformismo y herencia. Pero no te asustes con esos términos, ellos son abordados con un proyecto práctico que junto con nuestra metodología y didáctica hacen que el contenido sea mucho más fácil de ser aprendido.

ARTÍCULOS DE TECNOLOGÍA > PROGRAMACIÓN

**En Alura encontrarás variados cursos sobre Programación.
¡Comienza ahora!**

SEMESTRAL

US\$49,90

un solo pago de US\$49,90

- ✓ 218 cursos
- ✓ Videos y actividades 100% en Español
- ✓ Certificado de participación
- ✓ Estudia las 24 horas, los 7 días de la semana
- ✓ Foro y comunidad exclusiva para resolver tus dudas



Acceso a todo el contenido de la plataforma por 6 meses

¡QUIERO EMPEZAR A ESTUDIAR!

[Paga en moneda local en los siguientes países](#)

ANUAL

US\$79,90

un solo pago de US\$79,90



218 cursos



Videos y actividades 100% en Español



Certificado de participación



Estudia las 24 horas, los 7 días de la semana



Foro y comunidad exclusiva para resolver tus dudas



Acceso a todo el contenido de la plataforma por 12 meses

¡QUIERO EMPEZAR A ESTUDIAR!

[Paga en moneda local en los siguientes países](#)

Acceso a todos
los cursos

Estudia las 24 horas,
dónde y cuándo quieras

Nuevos cursos
cada semana

NAVEGACIÓN

PLANES
INSTRUCTORES
BLOG
POLÍTICA DE PRIVACIDAD
TÉRMINOS DE USO
SOBRE NOSOTROS
PREGUNTAS FRECUENTES

¡CONTÁCTANOS!

¡QUIERO ENTRAR EN CONTACTO!

BLOG

PROGRAMACIÓN
FRONT END
DATA SCIENCE
INNOVACIÓN Y GESTIÓN
DEVOPS

AOVS Sistemas de Informática S.A
CNPJ 05.555.382/0001-33

SÍGUENOS EN NUESTRAS REDES SOCIALES



ALIADOS



En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.



Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

CURSOS

Cursos de Programación

Lógica de Programación | Java

Cursos de Front End

HTML y CSS | JavaScript | React

Cursos de Data Science

Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps

Docker | Linux

Cursos de Innovación y Gestión

Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics | Liderazgo y Gestión de Equipos | Startups y Emprendimiento